Universidade Federal do ABC
Fundamentos de Processamento Gráfico 2023.3
Prof. Celso Setsuo Kurashima
Andressa Guimaraes Benedicto 11201810280
Heitor Rodrigues Savegnago 11077415
Kaleb Lucas da Silva Alves 21049916
Pedro Domingos Napole Certo 11201722682

Laboratório 05 - Cores e Mistura

1-O formato usado no comando glColor3f(r,g,b).
No código, cada um dos vértices tem uma cor específica, assim na hora que renderizamos, o resultado é um triângulo colorido, em que as diferentes cores do vértice se misturam de maneira suave, através de um gradiente dentro do triângulo.

2- O valor de alpha, tanto no triângulo esquerdo, quanto no direito, é 0.75.
Os parâmetros passados na função de mistura são GL_SRC_ALPHA e GL_ONE_MINUS_SRC_ALPHA.
No momento da mistura o código vai levar em consideração o valor alpha para saber a proporção de mistura que será aplicada . O código, então, compara a cor do objeto que está sendo modelado com a cor atual da imagem, pixel por pixel e, usando o valor de alpha, esse pixel será colorido como resultado da mistura das duas cores.

Programas elaborados em aula

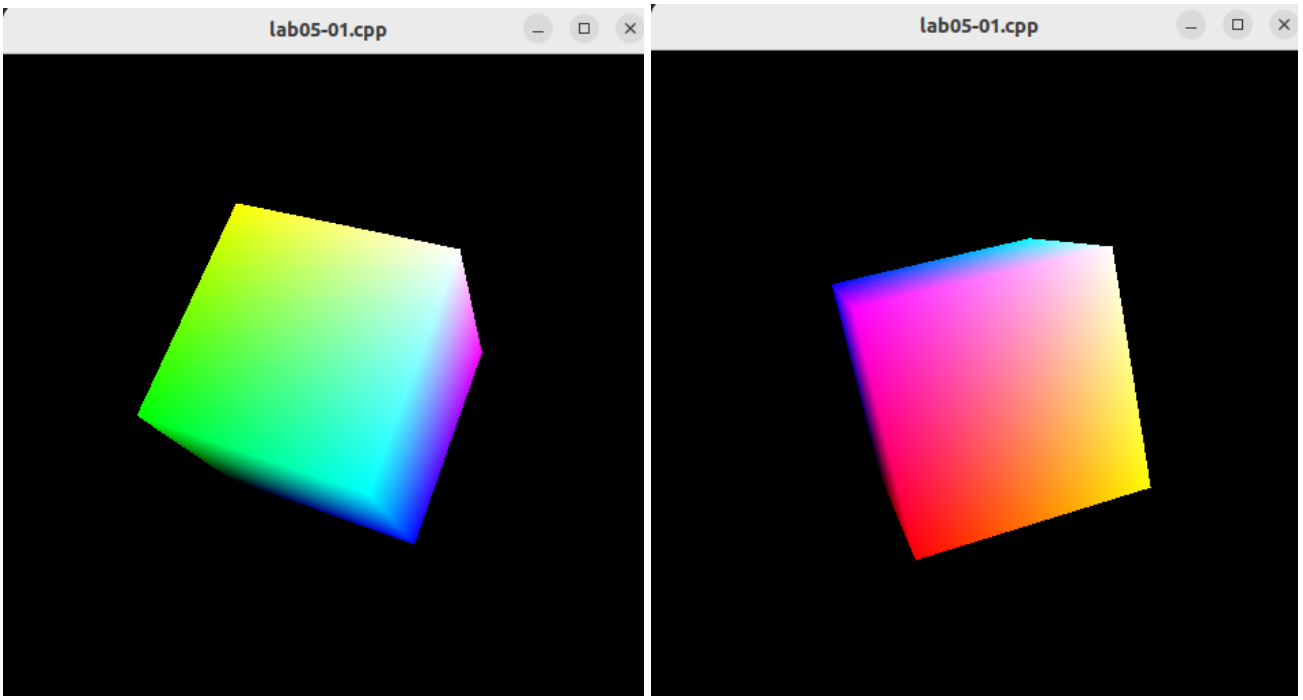
1-

```
/*
 * lab05-01.cpp
  * gcc lab02-02.cpp -o /tmp/temp.run -lm -lglut -lGL -lGLU && /tmp/temp.run
 */
#include <GL/glut.h>    // Header File For The GLUT Library
#include <GL/gl.h>    // Header File For The OpenGL32 Library
#include <GL/glu.h>    // Header File For The GLu32 Library
#include <unistd.h>       // needed to sleep

#define ESCAPE 27

int window;

float rtri = 0.0f;

float rquad = 0.0f;

void InitGL(int Width, int Height)          // We call this right after our OpenGL window is created.
{
  glClearColor(0.0f, 0.0f, 0.0f, 0.0f);       // This Will Clear The Background Color To Black
  glClearDepth(1.0);                          // Enables Clearing Of The Depth Buffer
  glDepthFunc(GL_LESS);                               // The Type Of Depth Test To Do
  glEnable(GL_DEPTH_TEST);                            // Enables Depth Testing
  glShadeModel(GL_SMOOTH);                             // Enables Smooth Color Shading

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();                           // Reset The Projection Matrix
```

```
  gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);    // Calculate The Aspect Ratio Of The
Window

  glMatrixMode(GL_MODELVIEW);
}

void ReSizeGLScene(int Width, int Height)
{
  if (Height==0)                                // Prevent A Divide By Zero If The Window Is Too Small
          Height=1;

  glViewport(0, 0, Width, Height);          // Reset The Current Viewport And Perspective Transformation

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

  gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
  glMatrixMode(GL_MODELVIEW);
}

void DrawGLScene()
{

  glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);    // Clear The Screen And The Depth
Buffer
  glLoadIdentity();                             // Reset The View

  glTranslatef(0.0f,0.0f,-7.0f);      // Move Right 3 Units, and back into the screen 7

  glRotatef(rquad,1.0f,1.0f,1.0f);          // Rotate The Cube On X, Y, and Z

  glBegin(GL_QUADS);                                // start drawing the cube.

  // top of cube
  glColor3f( 1.0f, 1.0f, 0.0f); glVertex3f( 1.0f, 1.0f,-1.0f);          // Top Right Of The Quad (Top)
  glColor3f( 0.0f, 1.0f, 0.0f); glVertex3f(-1.0f, 1.0f,-1.0f);          // Top Left Of The Quad (Top)
  glColor3f( 0.0f, 1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);          // Bottom Left Of The Quad (Top)
  glColor3f( 1.0f, 1.0f, 1.0f); glVertex3f( 1.0f, 1.0f, 1.0f);          // Bottom Right Of The Quad (Top)

  // bottom of cube
  glColor3f( 1.0f, 0.0f, 1.0f); glVertex3f( 1.0f,-1.0f, 1.0f);          // Top Right Of The Quad (Bottom)
  glColor3f( 0.0f, 0.0f, 1.0f); glVertex3f(-1.0f,-1.0f, 1.0f);          // Top Left Of The Quad (Bottom)
  glColor3f( 0.0f, 0.0f, 0.0f); glVertex3f(-1.0f,-1.0f,-1.0f);          // Bottom Left Of The Quad (Bottom)
  glColor3f( 1.0f, 0.0f, 0.0f); glVertex3f( 1.0f,-1.0f,-1.0f);          // Bottom Right Of The Quad (Bottom)

  // front of cube
  glColor3f( 1.0f, 1.0f, 1.0f); glVertex3f( 1.0f, 1.0f, 1.0f);          // Top Right Of The Quad (Front)
  glColor3f( 0.0f, 1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);          // Top Left Of The Quad (Front)
  glColor3f( 0.0f, 0.0f, 1.0f); glVertex3f(-1.0f,-1.0f, 1.0f);          // Bottom Left Of The Quad (Front)
  glColor3f( 1.0f, 0.0f, 1.0f); glVertex3f( 1.0f,-1.0f, 1.0f);          // Bottom Right Of The Quad (Front)

  // back of cube.
```

```
  glColor3f( 1.0f, 0.0f, 0.0f); glVertex3f( 1.0f,-1.0f,-1.0f);          // Top Right Of The Quad (Back)
  glColor3f( 0.0f, 0.0f, 0.0f); glVertex3f(-1.0f,-1.0f,-1.0f);          // Top Left Of The Quad (Back)
  glColor3f( 0.0f, 1.0f, 0.0f); glVertex3f(-1.0f, 1.0f,-1.0f);          // Bottom Left Of The Quad (Back)
  glColor3f( 1.0f, 1.0f, 0.0f); glVertex3f( 1.0f, 1.0f,-1.0f);          // Bottom Right Of The Quad (Back)

  // left of cube
  glColor3f( 0.0f, 1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);          // Top Right Of The Quad (Left)
  glColor3f( 0.0f, 1.0f, 0.0f); glVertex3f(-1.0f, 1.0f,-1.0f);          // Top Left Of The Quad (Left)
  glColor3f( 0.0f, 0.0f, 0.0f); glVertex3f(-1.0f,-1.0f,-1.0f);          // Bottom Left Of The Quad (Left)
  glColor3f( 0.0f, 0.0f, 1.0f); glVertex3f(-1.0f,-1.0f, 1.0f);          // Bottom Right Of The Quad (Left)

  // Right of cube
  glColor3f( 1.0f, 1.0f, 0.0f); glVertex3f( 1.0f, 1.0f,-1.0f);          // Top Right Of The Quad (Right)
  glColor3f( 1.0f, 1.0f, 1.0f); glVertex3f( 1.0f, 1.0f, 1.0f);          // Top Left Of The Quad (Right)
  glColor3f( 1.0f, 0.0f, 1.0f); glVertex3f( 1.0f,-1.0f, 1.0f);          // Bottom Left Of The Quad (Right)
  glColor3f( 1.0f, 0.0f, 0.0f); glVertex3f( 1.0f,-1.0f,-1.0f);          // Bottom Right Of The Quad (Right)
  glEnd();                                  // Done Drawing The Cube

  rtri  += 0.05f;                                   // Increase The Rotation Variable For The Pyramid
  rquad -= 0.05f;                                   // Decrease The Rotation Variable For The Cube

  // swap the buffers to display, since double buffering is used.
  glutSwapBuffers();
}

void keyPressed(unsigned char key, int x, int y)
{
        /* avoid thrashing this call */
        usleep(100);

        /* If escape is pressed, kill everything. */
        if (key == ESCAPE)
        {
        /* shut down our window */
        glutDestroyWindow(window);

        /* exit the program...normal termination. */
        exit(0);
        }
}

int main(int argc, char **argv)
{

  glutInit(&argc, argv);

  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH);
  glutInitWindowSize(480, 480);
  glutInitWindowPosition(100, 100);

  window = glutCreateWindow("lab05-01.cpp");

  glutDisplayFunc(&DrawGLScene);
```

```
    glutIdleFunc(&DrawGLScene);
    glutReshapeFunc(&ReSizeGLScene);
    glutKeyboardFunc(&keyPressed);
    InitGL(640, 480);
    glutMainLoop();

    return 1;
}
```



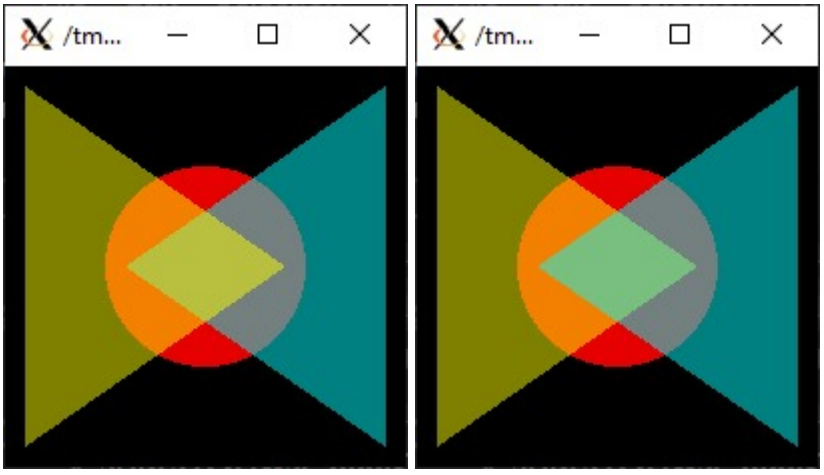
2-

```
/*
 * Copyright (c) 1993-1997, Silicon Graphics, Inc.
 * ALL RIGHTS RESERVED
 * Permission to use, copy, modify, and distribute this software for
 * any purpose and without fee is hereby granted, provided that the above
 * copyright notice appear in all copies and that both the copyright notice
 * and this permission notice appear in supporting documentation, and that
 * the name of Silicon Graphics, Inc. not be used in advertising
 * or publicity pertaining to distribution of the software without specific,
 * written prior permission.
 *
 * THE MATERIAL EMBODIED ON THIS SOFTWARE IS PROVIDED TO YOU "AS-IS"
 * AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE,
 * INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR
 * FITNESS FOR A PARTICULAR PURPOSE.  IN NO EVENT SHALL SILICON
 * GRAPHICS, INC.  BE LIABLE TO YOU OR ANYONE ELSE FOR ANY DIRECT,
 * SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY
 * KIND, OR ANY DAMAGES WHATSOEVER, INCLUDING WITHOUT LIMITATION,
 * LOSS OF PROFIT, LOSS OF USE, SAVINGS OR REVENUE, OR THE CLAIMS OF
 * THIRD PARTIES, WHETHER OR NOT SILICON GRAPHICS, INC.  HAS BEEN
 * ADVISED OF THE POSSIBILITY OF SUCH LOSS, HOWEVER CAUSED AND ON
 * ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE
 * POSSESSION, USE OR PERFORMANCE OF THIS SOFTWARE.
 *
 * US Government Users Restricted Rights
```

```
 * Use, duplication, or disclosure by the Government is subject to
 * restrictions set forth in FAR 52.227.19(c)(2) or subparagraph
 * (c)(1)(ii) of the Rights in Technical Data and Computer Software
 * clause at DFARS 252.227-7013 and/or in similar or successor
 * clauses in the FAR or the DOD or NASA FAR Supplement.
 * Unpublished-- rights reserved under the copyright laws of the
 * United States.  Contractor/manufacturer is Silicon Graphics,
 * Inc., 2011 N.  Shoreline Blvd., Mountain View, CA 94039-7311.
 *
 * OpenGL(R) is a registered trademark of Silicon Graphics, Inc.
 */

/*
 *  alpha.c
 *  This program draws several overlapping filled polygons
 *  to demonstrate the effect order has on alpha blending results.
 *  Use the 't' key to toggle the order of drawing polygons.
 */
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>

GLdouble PI = 3.1415926535897;
GLint circle_points = 100;
GLdouble radius = 0.5;

static int leftFirst = GL_TRUE;

/*  Initialize alpha blending function.
 */
static void init(void)
{
   glEnable (GL_BLEND);
   glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
   glShadeModel (GL_FLAT);
   glClearColor (0.0, 0.0, 0.0, 0.0);
}

static void drawCircle(void){
    glBegin(GL_POLYGON); // Circulo
        glColor4f(1.0, 0.0, 0.0, 0.9);
        for (int i = 0; i < circle_points; i++)
        {
                float angle = 2 * PI * i / circle_points;
                glVertex2f(radius*cos(angle), radius*sin(angle));
        }
    glEnd();
}

static void drawLeftTriangle(void)
{
```

```
    /* draw yellow triangle on LHS of screen */

    glBegin (GL_TRIANGLES);
        glColor4f(1.0, 1.0, 0.0, 0.5);
        glVertex3f(-0.9, 0.9, 0.0);
        glVertex3f(0.4, 0.0, 0.0);
        glVertex3f(-0.9, -0.9, 0.0);
    glEnd();
}

static void drawRightTriangle(void)
{
    /* draw cyan triangle on RHS of screen */

    glBegin (GL_TRIANGLES);
        glColor4f(0.0, 1.0, 1.0, 0.5);
        glVertex3f(0.9, 0.9, 0.0);
        glVertex3f(-0.4, 0.0, 0.0);
        glVertex3f(0.9, -0.9, 0.0);
    glEnd();
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    drawCircle();
    if (leftFirst) {
        drawLeftTriangle();
        drawRightTriangle();
    }
    else {
        drawRightTriangle();
        drawLeftTriangle();
    }

    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        gluOrtho2D (-1.0, 1.0, -1.0, 1.0*(GLfloat)h/(GLfloat)w);
    else
        gluOrtho2D (-1.0, 1.0*(GLfloat)w/(GLfloat)h, -1.0, 1.0);
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
```

```
        case 't':
        case 'T':
                leftFirst = !leftFirst;
                glutPostRedisplay();
                break;
        case 27:  /*  Escape key  */
                exit(0);
                break;
        default:
                break;
    }
}
```



Observação: A segunda imagem é referente à versão do código que está acima, em que as cores foram levemente alteradas para que houvesse uma melhor visualização da transparência.

Referências:

- NEIDER, Jackie; DAVIS, Tom; WOO, Mason. **OpenGL programming guide**. Reading, MA: Addison-Wesley, 1993.